In the Claims:

Please amend Claims 1-3 and 8-10; cancel Claims 7 and 14; and add new Claims 15-21, all as shown below. Applicant respectfully reserves the right to prosecute any originally presented claims in a continuing or future application.

1. (Currently Amended) A system including a command-line interface for use with a [[JMS]] mark-up language, comprising:

a computer including a processing device and a client operating thereon;

an application including a command-line user interface that executes on ~~a client machine~~ the client and allows a user to enter markup language ~~components~~ commands;

a command processor that executes on the client and that validates the markup language commands, and, for each markup language command converts the markup language command into a command object for communication to a command dispatcher;

a command dispatcher that executes on the client and that receives command objects from the command processor and, for each command object, assigns the command object to one of a plurality of categories corresponding to a plurality of application program interfaces; and

a plurality of processor modules that execute on the client, including a processor module for each category of application program interface, wherein each processor module receives the command objects assigned to its category, and performs appropriate operations against the corresponding application program interface ~~a command processor that converts the markup language components into one of JMS or JMX system operations and executes said JMS or JMX system operations~~ at a remote server.

2. (Currently Amended) The system of claim 1 wherein the markup language ~~components~~ commands are communicated as a source file, and wherein the client includes a parser that parses said source file to retrieve said markup language ~~components~~ commands and communicate said markup language ~~components~~ commands to said server.

3. (Currently Amended) The system of claim 1 wherein said command-line user interface communicates said markup language ~~components~~ commands to said remote server via a wide area network or the Internet.

4. (Original) The system of claim 1 wherein said parser and said command processor comprise an engine that parses source files and generates commands.

5. (Original) The system of claim 1 wherein the markup language is JMS markup language.

6. (Original) The system of claim 1 wherein the source file is an XML file.

7. (Canceled).

8. (Currently Amended) A method of using a command-line interface with a [[JMS]] mark-up language, comprising the steps of:

providing an application including a command-line user interface that executes on a client ~~machine~~ and allows a user to enter markup language ~~components~~ commands;

receiving said markup language ~~components~~ commands at a command processor that executes on the client and that validates the markup language commands, and, for each markup language command converts the markup language command into a command object;

assigning each command object to one of a plurality of categories corresponding to a plurality of application program interfaces; and

~~converting the markup language into one of JMS or JMX system operations~~

processing the command objects using a plurality of processor modules, including a processor module for each category of application program interface, wherein each processor module receives the command objects assigned to its category, and performs appropriate operations against the corresponding application program interface at a remote server.

9. (Currently Amended) The method of claim 8 wherein the markup language ~~components~~ commands are communicated as a source file, and wherein the client includes a parser that parses

said source file to retrieve said markup language ~~components~~ commands and communicate said markup language ~~components~~ commands to said command processor.

10. (Currently Amended) The method of claim 8 wherein said command-line user interface communicates said markup language ~~components~~ commands to said remote server via a wide area network or the Internet.

11. (Original) The method of claim 8 wherein said parser and said command processor comprise an engine that parses source files and generates commands.

12. (Original) The method of claim 8 wherein the markup language is JMS markup language.

13. (Original) The method of claim 8 wherein the source file is an XML file.

14. (Canceled).

15. (New) The system of claim 1 wherein at least one of the application program interfaces conforms to the Java Message Service specification.

16. (New) The system of claim 1 wherein at least one of the application program interfaces conforms to the Java Management Extensions specification.

17. (New) The system of claim 1 wherein the plurality of application program interfaces include both application program interface that conforms to the Java Message Service specification and an application program interface that conforms to the Java Management Extensions specification.

18. (New) The method of claim 8 wherein at least one of the application program interfaces conforms to the Java Message Service specification.

19. (New) The method of claim 8 wherein at least one of the application program interfaces conforms to the Java Management Extensions specification.

20. (New) The method of claim 8 wherein the plurality of application program interfaces include both application program interface that conforms to the Java Message Service specification and an application program interface that conforms to the Java Management Extensions specification.

21. (New) A computer readable medium including instructions stored thereon, which when executed cause the computer to perform the steps of:

providing an application including a command-line user interface that executes on a client and allows a user to enter markup language commands;

receiving said markup language commands at a command processor that executes on the client and that validates the markup language commands, and, for each markup language command converts the markup language command into a command object;

assigning each command object to one of a plurality of categories corresponding to a plurality of application program interfaces; and

processing the command objects using a plurality of processor modules, including a processor module for each category of application program interface, wherein each processor module receives the command objects assigned to its category, and performs appropriate operations against the corresponding application program interface at a remote server.